UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/646,309	08/22/2003	Gregory M. Wright	SUN-P9042-SPL	9198

57960 7590 05/22/2008
PVF -- SUN MICROSYSTEMS INC.
C/O PARK, VAUGHAN & FLEMING LLP
2820 FIFTH STREET
DAVIS, CA 95618-7759

EXAMINER	
CHEN, QING	
ART UNIT	**PAPER NUMBER**
2191	

MAIL DATE	DELIVERY MODE
05/22/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)
	10/646,309	WRIGHT ET AL.
	Examiner	Art Unit
	Qing Chen	2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 February 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18 and 28-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18 and 28-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the RCE filed on February 13, 2008.

2. **Claims 1-18 and 28-35** are pending.

3. **Claims 1, 10-18, 28, and 32-35** have been amended.

4. **Claims 19-27 and 36-39** have been cancelled.

5. The 35 U.S.C. § 101 rejections of Claims 10-18 and 32-35 are withdrawn in view of Applicant's amendments to the claims.

Response to Amendment

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. **Claims 1-7, 10-16, and 28-35** are rejected under 35 U.S.C. 103(a) as being unpatentable over **US 6,289,506 (hereinafter "Kwong")** in view of **US 7,032,216 (hereinafter "Nizhegorodov")**.

As per **Claim 1**, Kwong discloses:

- selecting a call to a native code method to be optimized within the virtual machine *(see Figure 7: 730; Column 8: 32-35, "... if the programmer decides to try to improve*

performance, then at step 730, he may select some of the Java program methods on the candidate list from step 720 for optimization.");

- decompiling at least part of the native code method into an intermediate representation *(see Figure 7: 735; Column 8: 38-43, "... a user may decide to de-compile earlier native compiled code back to bytecode format. The de-compile process may be used for instance when a user determines that the native compiled code does not present the desired performance and the user wants to revert the native compiled code back to Java bytecode.")*;

- obtaining an intermediate representation associated with the application running on the virtual machine which interacts with the native code method *(see Figure 7: 705; Column 8: 23-25, "A programmer would first write a computer program in the Java programming language in step 705." Note that the source code is modified after an iteration of the optimization loop.)*;

- combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine to form a combined intermediate representation *(see Figure 7: 705 and 740 (Note that the DLL for the native methods are incorporated into the source code.); Column 8: 35-38, "... the selected Java program methods are optimized and compiled into native processor code by a native Java compiler."; Column 10: 8-10, "... the Java application now comprises of Lib.dll 1060, A.class 1010, and B.class 1020 and may be executed on a Java VM 1080.")*; and

- generating native code from the combined intermediate representation, wherein the native code generation process optimizes interactions between the application running on the virtual machine and the native code method *(see Figure 7: 710, 715, and 720; Column 8: 46-47,*

"… a programmer may repeat these steps to further refine and optimize the program."; Column 9: 9-11, "Once the bytecodes are in the Java VM 840, they are interpreted by a Java interpreter 842 or turned into native machine code by the JIT compiler 844.").

However, Kwong does not disclose:

- wherein combining the intermediate representation involves inlining native code methods into call sites in the application.

Nizhegorodov discloses:

- wherein combining the intermediate representation involves inlining native code methods into call sites in the application *(see Column 5: 28-31, "Alternatively, the native compiler can also inline small methods in place of monomorphic call sites, triggering further optimizations and code improvements.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Nizhegorodov into the teaching of Kwong to include wherein combining the intermediate representation involves inlining native code methods into call sites in the application. The modification would be obvious because one of ordinary skill in the art would be motivated to improve code optimizations *(see Nizhegorodov – Column 5: 28-31).*

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Kwong further discloses:

- wherein selecting the call to the native code method involves selecting the call based upon at least one of: the execution frequency of the call *(see Column 4: 11-19, "An analysis tool may track the Java program methods entered and exited in memory, establish a relationship*

between parent and child methods called, record every called program method, and time spend in each method. In another embodiment, an analysis tool may keep track of the Java methods being loaded into memory along with active software executing on the system. A tuning tool may determine the most active classes and methods in a Java application and list possible candidates for native compilation."); and the overhead involved in performing the call to the native code method as compared against the amount of work performed by the native code method.

As per **Claim 3**, the rejection of **Claim 1** is incorporated; and Kwong further discloses:

- wherein optimizing interactions between the application running on the virtual machine and the native code method involves optimizing calls to the native code method by the application *(see Column 8: 32-35, "... if the programmer decides to try to improve performance, then at step 730, he may select some of the Java program methods on the candidate list from step 720 for optimization.")*.

As per **Claim 4**, the rejection of **Claim 1** is incorporated; and Kwong further discloses:

- wherein optimizing interactions between the application running on the virtual machine and the native code method involves optimizing callbacks by the native code method into the virtual machine *(see Column 7: 9-12, "In order to maintain the state of the Java VM 430 and make system calls, the compiled Java code 440 may make calls 450 into the Java VM 430.")*.

As per **Claim 5**, the rejection of **Claim 4** is incorporated; however, Kwong does not disclose:

- wherein optimizing callbacks by the native code method into the virtual machine involves optimizing callbacks that access heap objects within the virtual machine.

Official Notice is taken that it is old and well-known within the computing art to allow callbacks to access heap objects within the virtual machine. Applicant has submitted in the specification that JNI™ provides an interface through which native code can manipulate heap objects within the JVM™ in a platform-independent way *(see Page 2, Paragraph [0004])*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include wherein optimizing callbacks by the native code method into the virtual machine involves optimizing callbacks that access heap objects within the virtual machine. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the implementation of a Java™ object to remain transparent to the native code.

As per **Claim 6**, the rejection of **Claim 4** is incorporated; and Kwong further discloses:

- wherein the virtual machine is a platform-independent virtual machine *(see Figure 2: 212)*; and

- wherein combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine involves integrating calls provided by an interface for accessing native code into the native code method *(see Column 5: 41-44, "A Java Native Interface (JNI) may exist with the Java VM 212. The Java Native Interface is a standard programming interface for writing Java native methods and embedding the Java VM into native applications."; Column 10: 10-13, "When the method in*

A.class 1010 is native compiled, it needs to use the Java native interface 1070 to access the field b in class B 1020.").

As per **Claim 7**, the rejection of **Claim 1** is incorporated; and Kwong further discloses:

- wherein obtaining the intermediate representation associated with the application running on the virtual machine involves recompiling a corresponding portion of the application *(see Column 8: 48-50, "The process of monitoring and compiling bytecode/de-compiling native code may be repeated until the desired performance is obtained.").*

Claims 10-16 are computer-readable storage device claims corresponding to the method claims above (Claims 1-7) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1-7.

As per **Claim 28**, Kwong discloses:

- deciding to optimize a callback by a native code method into the virtual machine *(see Figure 7: 730; Column 7: 9-12, "In order to maintain the state of the Java VM 430 and make system calls, the compiled Java code 440 may make calls 450 into the Java VM 430."; Column 8: 32-35, "... if the programmer decides to try to improve performance, then at step 730, he may select some of the Java program methods on the candidate list from step 720 for optimization.")*;
- decompiling at least part of the native code method into an intermediate representation *(see Figure 7: 735; Column 8: 38-43, "... a user may decide to de-compile earlier native compiled code back to bytecode format. The de-compile process may be used for instance*

when a user determines that the native compiled code does not present the desired performance and the user wants to revert the native compiled code back to Java bytecode.");

- obtaining an intermediate representation associated with the application running on the virtual machine which interacts with the native code method *(see Figure 7: 705; Column 8: 23-25, "A programmer would first write a computer program in the Java programming language in step 705." Note that the source code is modified after an iteration of the optimization loop.)*;

- combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine to form a combined intermediate representation *(see Figure 7: 705 and 740 (Note that the DLL for the native methods are incorporated into the source code.); Column 8: 35-38, "... the selected Java program methods are optimized and compiled into native processor code by a native Java compiler."; Column 10: 8-10, "... the Java application now comprises of Lib.dll 1060, A.class 1010, and B.class 1020 and may be executed on a Java VM 1080.")*; and

- generating native code from the combined intermediate representation, wherein the native code generation process optimizes the callback by the native code method into the virtual machine *(see Figure 7: 710, 715, and 720; Column 8: 46-47, "... a programmer may repeat these steps to further refine and optimize the program."; Column 9: 9-11, "Once the bytecodes are in the Java VM 840, they are interpreted by a Java interpreter 842 or turned into native machine code by the JIT compiler 844.").*

However, Kwong does not disclose:

- wherein combining the intermediate representation involves inlining native code methods into call sites in the application.

Nizhegorodov discloses:

- wherein combining the intermediate representation involves inlining native code methods into call sites in the application *(see Column 5: 28-31, "Alternatively, the native compiler can also inline small methods in place of monomorphic call sites, triggering further optimizations and code improvements.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Nizhegorodov into the teaching of Kwong to include wherein combining the intermediate representation involves inlining native code methods into call sites in the application. The modification would be obvious because one of ordinary skill in the art would be motivated to improve code optimizations *(see Nizhegorodov – Column 5: 28-31)*.

As per **Claim 29**, the rejection of **Claim 28** is incorporated; and Kwong further discloses:

- wherein the native code generation process also optimizes calls to the native code method by the application *(see Column 8: 32-35, "... if the programmer decides to try to improve performance, then at step 730, he may select some of the Java program methods on the candidate list from step 720 for optimization.")*.

As per **Claim 30**, the rejection of **Claim 28** is incorporated; however, Kwong does not disclose:

- wherein optimizing the callback by the native code method into the virtual machine involves optimizing a callback that accesses a heap object within the virtual machine.

Official Notice is taken that it is old and well-known within the computing art to allow callbacks to access heap objects within the virtual machine. Applicant has submitted in the specification that JNI™ provides an interface through which native code can manipulate heap objects within the JVM™ in a platform-independent way *(see Page 2, Paragraph [0004])*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include wherein optimizing the callback by the native code method into the virtual machine involves optimizing a callback that accesses a heap object within the virtual machine. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the implementation of a Java™ object to remain transparent to the native code.

As per **Claim 31**, the rejection of **Claim 28** is incorporated; and Kwong further discloses:

- wherein the virtual machine is a platform-independent virtual machine *(see Figure 2: 212)*; and

- wherein combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine involves integrating calls provided by an interface for accessing native code into the native code method *(see Column 5: 41-44, "A Java Native Interface (JNI) may exist with the Java VM 212. The Java Native Interface is a standard programming interface for writing Java native methods and embedding the Java VM into native applications."; Column 10: 10-13, "When the method in*

A.class 1010 is native compiled, it needs to use the Java native interface 1070 to access the field b in class B 1020.").

Claims 32-35 are computer-readable storage device claims corresponding to the method claims above (Claims 28-31) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 28-31.

8. **Claims 8 and 17** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Kwong** in view of **Nizhegorodov** as applied to Claims 1 and 10 above, and further in view of **US 5,491,821 (hereinafter "Kilis")**.

As per **Claim 8**, the rejection of **Claim 1** is incorporated; however, Kwong and Nizhegorodov do not disclose:

- wherein obtaining the intermediate representation associated the application running on the virtual machine involves accessing a previously generated intermediate representation associated with the application running on the virtual machine.

Kilis discloses:

- wherein obtaining the intermediate representation associated the application running on the virtual machine involves accessing a previously generated intermediate representation associated with the application running on the virtual machine *(see Column 2: 2-4, "If the selected changed facet affects the object itself, then the previous intermediate representation of the object is modified.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kilis into the teaching of Kwong to include wherein obtaining the intermediate representation associated the application running on the virtual machine involves accessing a previously generated intermediate representation associated with the application running on the virtual machine. The modification would be obvious because one of ordinary skill in the art would be motivated to not reprocess existing information *(see Kilis – Column 1: 40-43)*.

Claim 17 is rejected for the same reason set forth in the rejection of Claim 8.

9. **Claims 9 and 18** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Kwong** in view of **Nizhegorodov** as applied to Claims 1 and 10 above, and further in view of **US 5,805,899 (hereinafter "Evans")**.

As per **Claim 9**, the rejection of **Claim 1** is incorporated; and Kwong further discloses:

- determining a signature of the call to the native code method *(see Column 4: 11-19, "An analysis tool may track the Java program methods entered and exited in memory, establish a relationship between parent and child methods called, record every called program method, and time spend in each method. In another embodiment, an analysis tool may keep track of the Java methods being loaded into memory along with active software executing on the system. A tuning tool may determine the most active classes and methods in a Java application and list possible candidates for native compilation.")*.

However, Kwong and Nizhegorodov do not disclose:

- determining a mapping from arguments of the call to corresponding locations in a native application binary interface (ABI).

Evans discloses:

- determining a mapping from arguments of the call to corresponding locations in a native application binary interface (ABI) *(see Column 7: 29-33, "Shared object 114 provides global symbols to which other objects, such as dynamic executable 120, can bind at runtime. These global symbols are specified in mapfile 130 and describe an Application Binary Interface (ABI) of shared object 114.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Evans into the teaching of Kwong to include wherein determining a mapping from arguments of the call to corresponding locations in a native application binary interface (ABI). The modification would be obvious because one of ordinary skill in the art would be motivated to describe the low-level interface between an application program and the operating system, its libraries, or components of the application program.

Claim 18 is rejected for the same reason set forth in the rejection of Claim 9.

Response to Arguments

10. Applicant's arguments with respect to Claims 1, 10, 28, and 32 have been considered, but are moot in view of the new ground(s) of rejection.

Conclusion

11. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/QC/
March 26, 2008

/Wei Zhen/

Supervisory Patent Examiner, Art Unit 2191